DEVIEW 2020

내 앱에 네이버 지도 넣기
네이버 지도 안드로이드 SDK 적용 및 활용

정기용 NAVER

CONTENTS

1. 네이버 지도 안드로이드 SDK 소개

1.1 네이버 지도 모바일 SDK

- 풀 벡터 렌더링 모바일 지도 엔진
- 네이버 지도 앱에서 사용하는 것과 동일한 SDK
- 무제한/무료로 개방되어 있는 오픈 API
- 네이버 클라우드 플랫폼을 통한 고객지원 및 질의응답 제공

1.2 다양하고 강력한 기능

풀 벡터 렌더링

1.2 다양하고 강력한 기능

다양한 지도 유형

1.2 다양하고 강력한 기능

DEVIEW 2020

다양한 오버레이

1.3 쓰기 쉬운 API

DEVIEW 2020

현대적인 API

```
val cameraUpdate = CameraUpdate.scrollTo(LatLng(37.5666102, 126.9783881))
    .reason(3)
    .animate(CameraAnimation.Easing, 2000)
    .finishCallback {
        Toast.makeText(context, "완료", Toast.LENGTH_SHORT).show()
    }
    .cancelCallback {
        Toast.makeText(context, "취소", Toast.LENGTH_SHORT).show()
    }

naverMap.moveCamera(cameraUpdate)
```

1.3 쓰기 쉬운 API

DEVIEW 2020

직관적인 API

```
val infoWindow = InfoWindow().apply {
    setOnClickListener {
        close()
        true
    }
}

Marker().apply {
    position = LatLng(37.5670135, 126.9783740)
    map = naverMap
    setOnClickListener {
        infoWindow.open(this)
        true
    }
}
```

1.3 쓰기 쉬운 API

양질의 문서와 예제 코드

- 문서: https://navermaps.github.io/android-map-sdk
- 예제: https://github.com/navermaps/android-map-sdk

1.3 쓰기 쉬운 API

DEVIEW 2020

양질의 문서와 예제 코드

개발 가이드

- 시작하기
- 지도와 좌표
  - 지도 객체
  - 좌표 객체
  - 지도 옵션
- 카메라
  - 카메라와 투영
  - 카메라 이동
- 상호작용
  - 사용자 인터페이스
  - 위치
- 오버레이
  - 오버레이 공통
  - 마커
  - 정보 창
  - 셰이프
  - 위치 오버레이
  - 지상 오버레이
  - 경로선과 화살표

API 호출로 카메라 이동하기

카메라를 움직이려면 먼저 카메라를 어떻게 움직일지를 나타내는 `CameraUpdate` 객체를 생성해야 합니다. `CameraUpdate` 는 카메라를 이동할 위치, 방법 등을 정의하는 클래스입니다. `CameraUpdate` 객체는 오직 팩토리 메서드를 이용해서 생성할 수 있으며, `CameraUpdate` 클래스에 다양한 팩토리 메서드가 정의되어 있습니다. 다음은 몇 가지 팩토리 메서드에 대한 설명입니다.

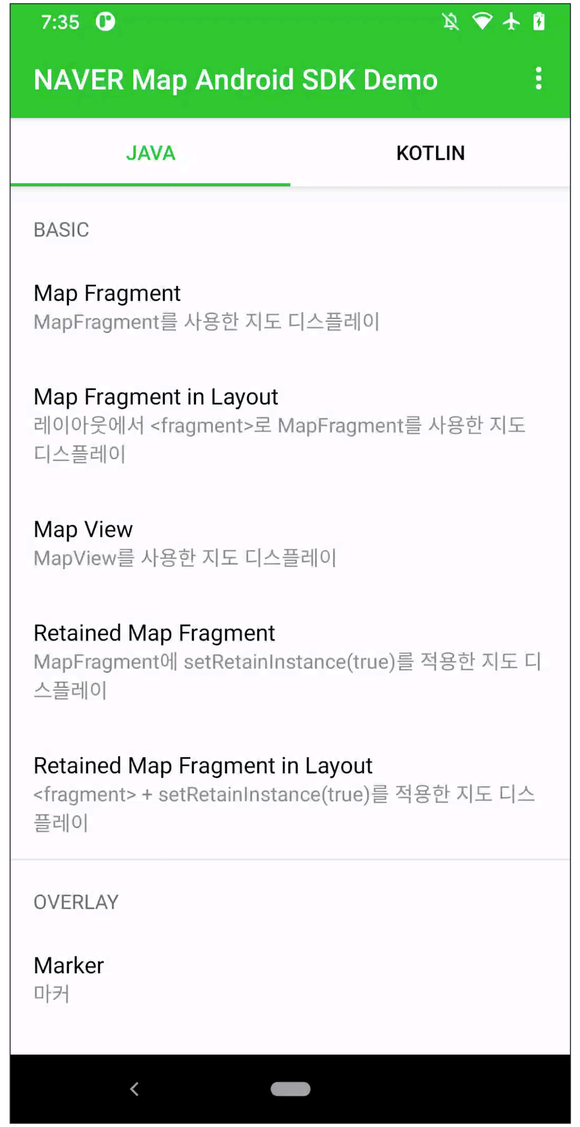
- `toCameraPosition()` : 카메라의 위치를 지정한 `CameraPosition` 으로 움직입니다.
- `scrollTo()` : 카메라의 대상 지점을 지정한 좌표로 변경합니다.
- `scrollBy()` : 카메라의 대상 지점을 지정한 픽셀만큼 상하좌우로 이동합니다.
- `zoomTo()` : 카메라의 줌 레벨을 지정한 값으로 변경합니다.
- `fitBounds()` : 영역이 온전히 보이는 좌표와 최대 줌 레벨로 카메라의 위치를 변경합니다.

생성한 객체를 파라미터로 삼아 `NaverMap.moveCamera()` 를 호출하면 카메라가 움직입니다.

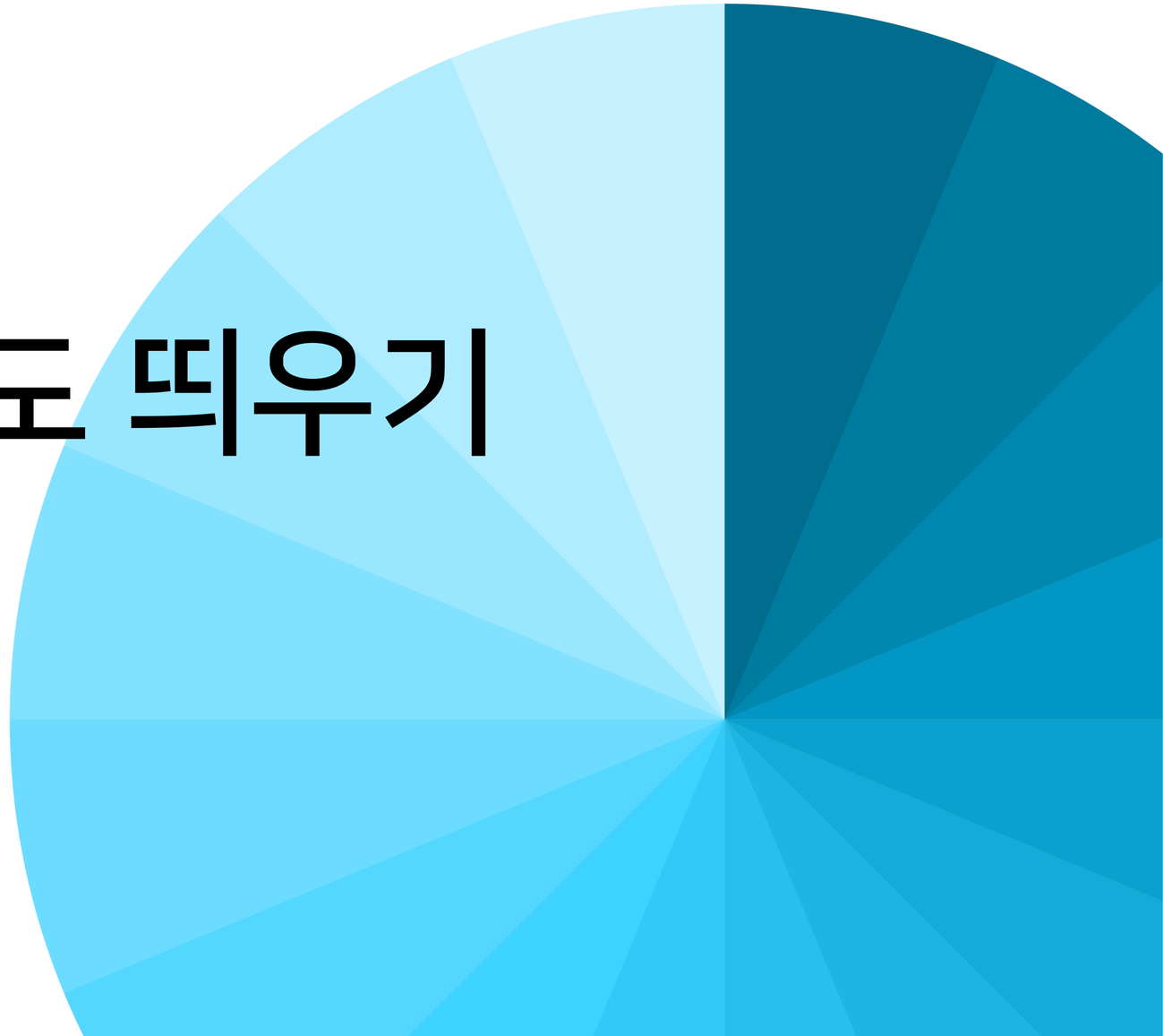
다음은 대상 지점을 `(37.5666102, 126.9783881)` 로 변경하도록 지정하는 `CameraUpdate` 객체를 만들고 `NaverMap.moveCamera()` 를 호출해 카메라를 움직이는 예제입니다.

Java | **Kotlin**

```
CameraUpdate cameraUpdate = CameraUpdate.scrollTo(new LatLng(37.5666102, 126.9783881));
naverMap.moveCamera(cameraUpdate);
```



2. 첫 지도 띄우기



2.1 네이버 클라우드 플랫폼 서비스 신청

DEVIEW 2020

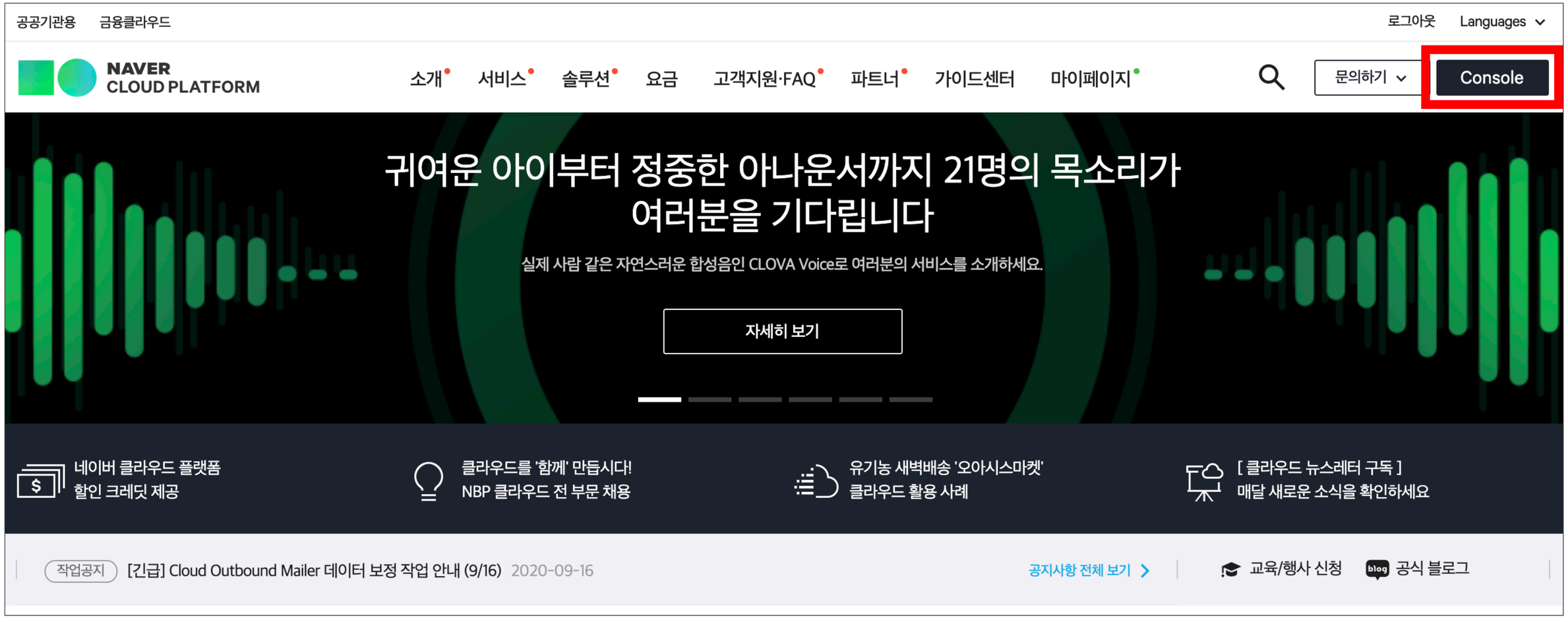
네이버 지도 안드로이드 SDK는 네이버 클라우드 플랫폼에서 제공

- https://www.ncloud.com
- 회원 가입 및 결제수단 등록 필요

2.1 네이버 클라우드 플랫폼 서비스 신청

DEVIEW 2020

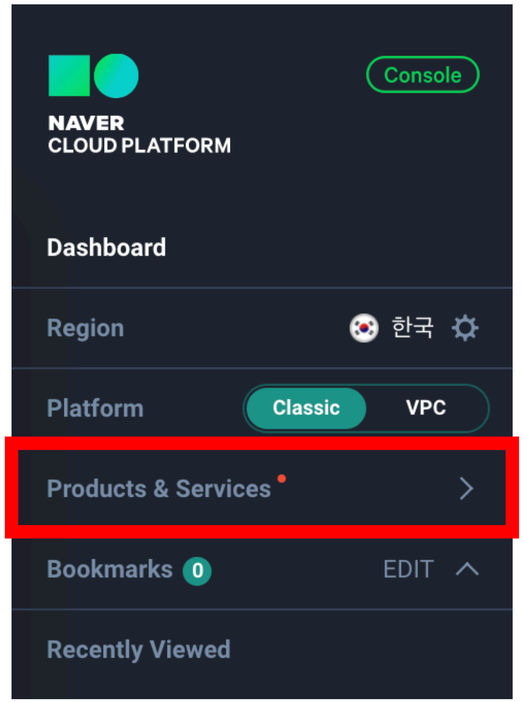
로그인 후 Console 선택



2.1 네이버 클라우드 플랫폼 서비스 신청

DEVIEW 2020

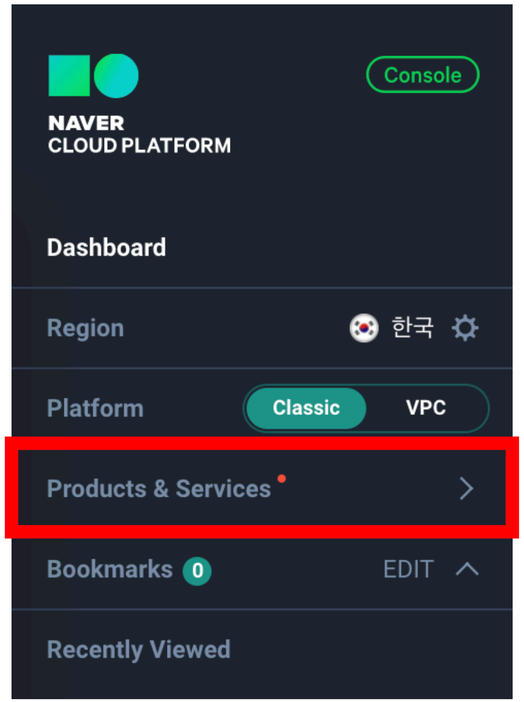
Products & Services → AI-Application Service → AI·NAVER API

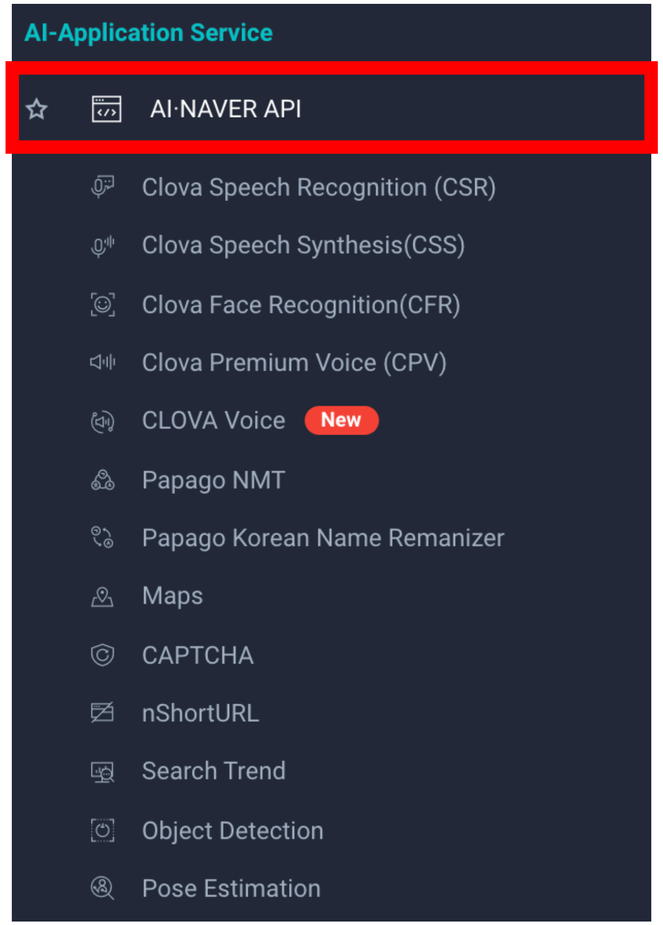


2.1 네이버 클라우드 플랫폼 서비스 신청

DEVIEW 2020

Products & Services → AI·Application Service → AI·NAVER API

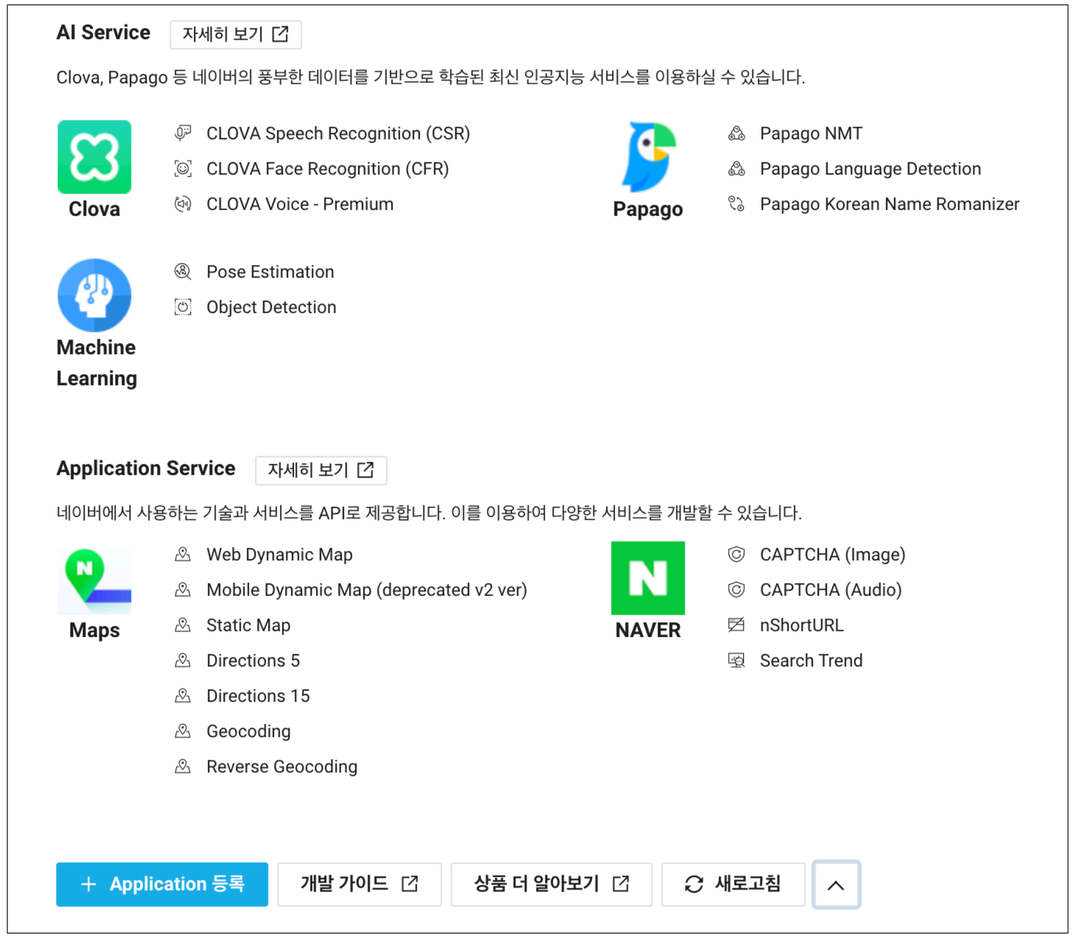




2.1 네이버 클라우드 플랫폼 서비스 신청

DEVIEW 2020

Application 등록 선택



2.1 네이버 클라우드 플랫폼 서비스 신청

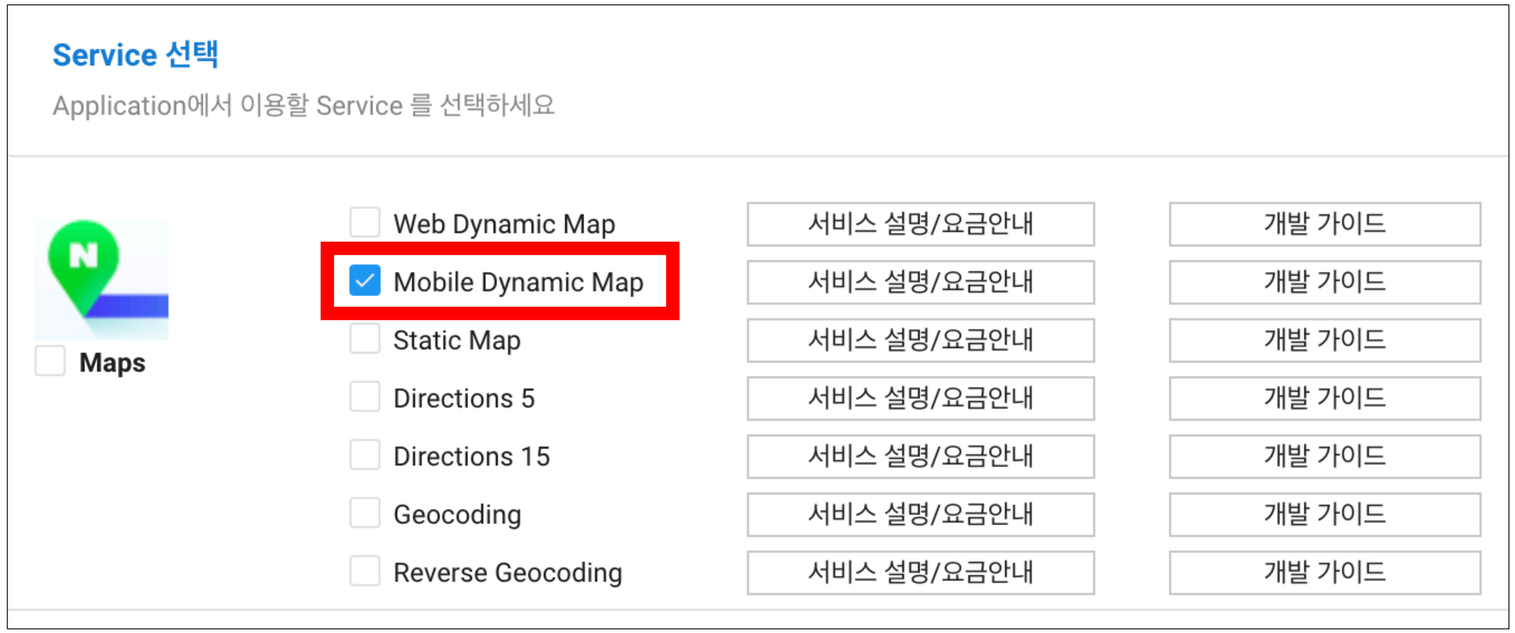
DEVIEW 2020

Application 이름: 임의의 이름 지정

2.1 네이버 클라우드 플랫폼 서비스 신청

DEVIEW 2020

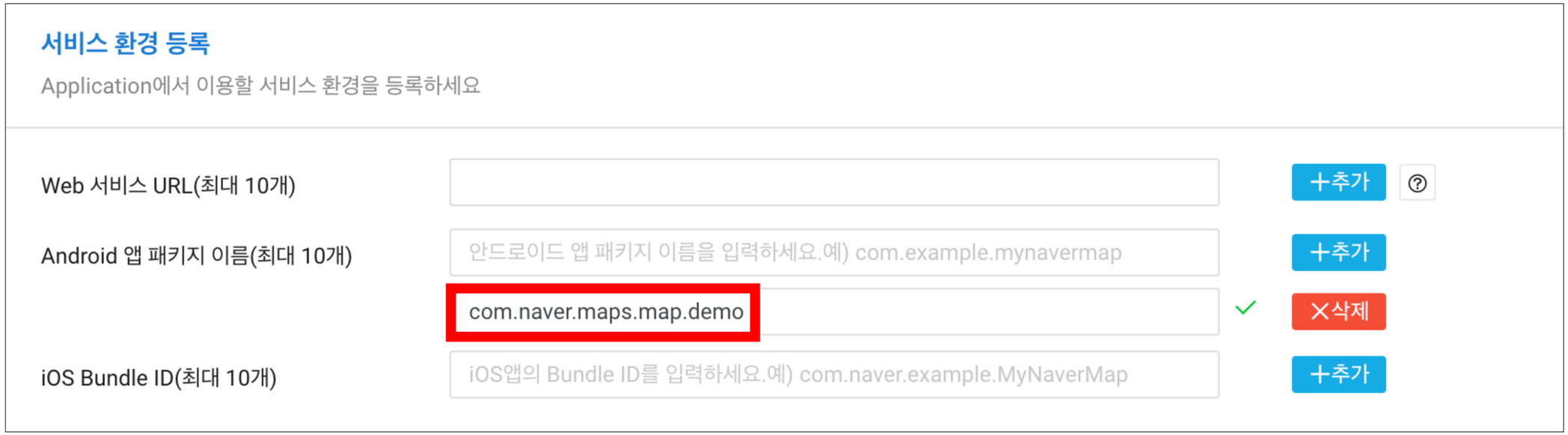
Maps → Mobile Dynamic Map 서비스 선택



2.1 네이버 클라우드 플랫폼 서비스 신청

DEVIEW 2020

Android 앱 패키지 이름에 com.naver.maps.map.demo 추가 후 등록



2.1 네이버 클라우드 플랫폼 서비스 신청

DEVIEW 2020

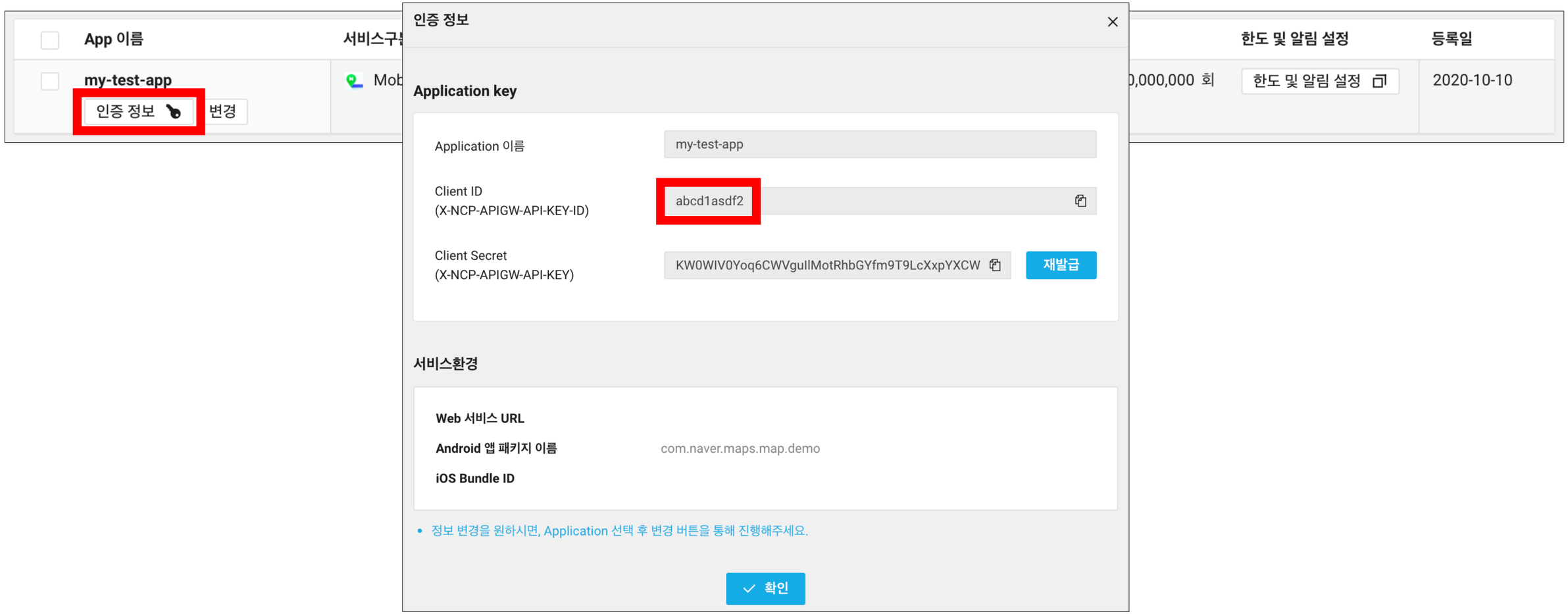
클라이언트 ID 확인

App 이름	서비스구분	당일 사용량	당월 사용량	한도 및 알림 설정	등록일
my-test-app 인증 정보 변경	Mobile Dynamic Map	0% 0/100,000,000 회	0% 0/100,000,000 회	한도 및 알림 설정	2020-10-10

2.1 네이버 클라우드 플랫폼 서비스 신청

DEVIEW 2020

클라이언트 ID 확인



2.2 데모 프로젝트 실행해보기

DEVIEW 2020

데모 프로젝트 클론

- https://github.com/navermaps/android-map-sdk

```
$ git clone https://github.com/navermaps/android-map-sdk
```

2.2 데모 프로젝트 실행해보기

DEVIEW 2020

client_id.xml에 발급받은 클라이언트 ID 입력

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="naver_map_sdk_client_id" translatable="false">YOUR_CLIENT_ID_HERE</string>
</resources>
```

2.2 데모 프로젝트 실행해보기

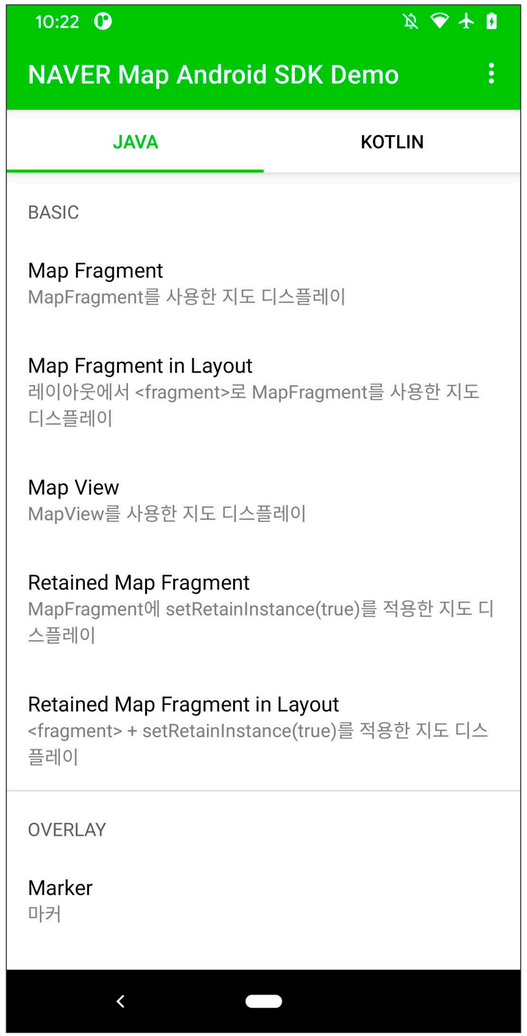
client_id.xml에 발급받은 클라이언트 ID 입력

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="naver_map_sdk_client_id" translatable="false">abcd1asdf2</string>
</resources>
```

2.2 데모 프로젝트 실행해보기

DEVIEW 2020

실행!



2.3 내 프로젝트에 지도 넣기

DEVIEW 2020

Hands-on: 함께 코딩하며 실습

- Kotlin
- AndroidX

2.3 내 프로젝트에 지도 넣기

DEVIEW 2020

안드로이드 프로젝트 준비 혹은 뼈대 프로젝트 클론

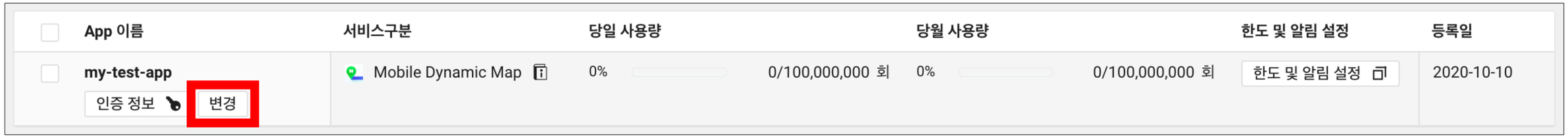
- https://github.com/themics/deview-2020-my-map

```
$ git clone https://github.com/themics/deview-2020-my-map
```

2.3 내 프로젝트에 지도 넣기

DEVIEW 2020

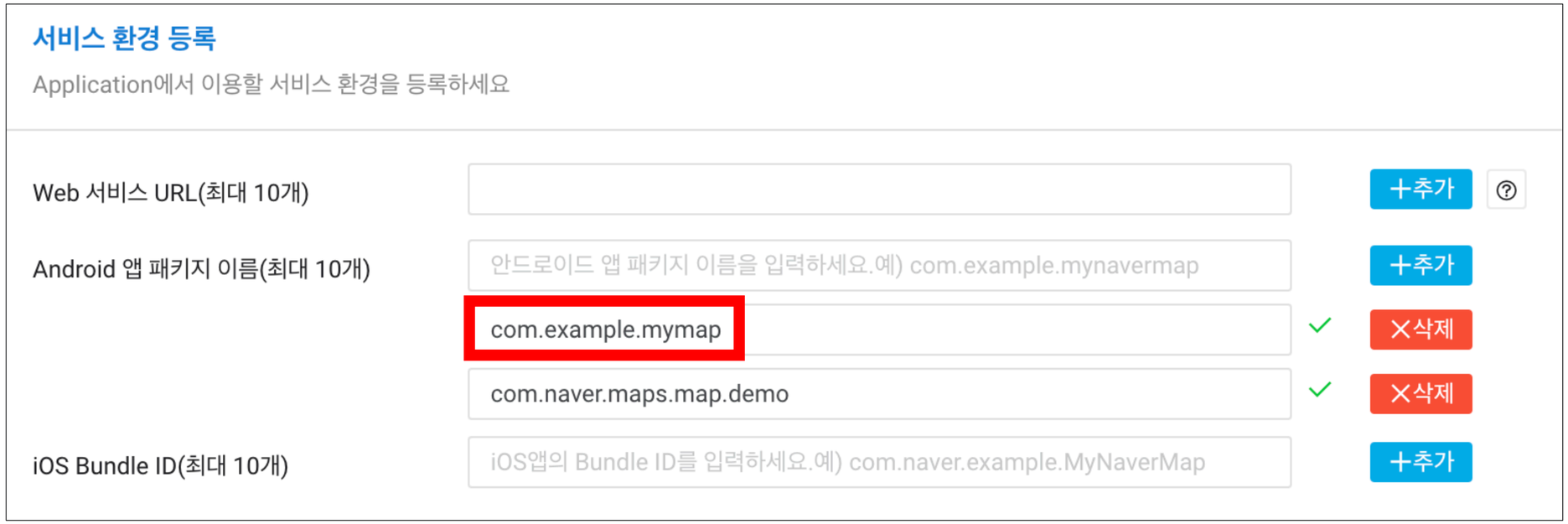
Android 앱 패키지 이름에 com.example.mymap 추가



2.3 내 프로젝트에 지도 넣기

DEVIEW 2020

Android 앱 패키지 이름에 com.example.mymap 추가



2.3 내 프로젝트에 지도 넣기

루트 프로젝트의 build.gradle에 저장소 추가

```
allprojects {
    repositories {
        google()
        jcenter()

        // 네이버 지도 안드로이드 저장소
        maven {
            url 'https://navercorp.bintray.com/maps'
        }
    }
}
```

2.3 내 프로젝트에 지도 넣기

앱 모듈의 build.gradle에 라이브러리 의존성 추가

```
dependencies {
    implementation 'com.naver.maps:map-sdk:3.10.0'
}
```

2.3 내 프로젝트에 지도 넣기

DEVIEW 2020

AndroidManifest.xml에 클라이언트 ID 메타데이터 추가

```
<application
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
    android:theme="@style/AppTheme">

    <meta-data
        android:name="com.naver.maps.map.CLIENT_ID"
        android:value="asdf1abcd2" />

    <!-- ... -->

</application>
```

2.3 내 프로젝트에 지도 넣기

DEVIEW 2020

레이아웃에 com.naver.maps.map.MapFragment 프래그먼트 추가

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <androidx.fragment.app.FragmentContainerView
        android:id="@+id/map_fragment"
        android:name="com.naver.maps.map.MapFragment"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

2.3 내 프로젝트에 지도 넣기

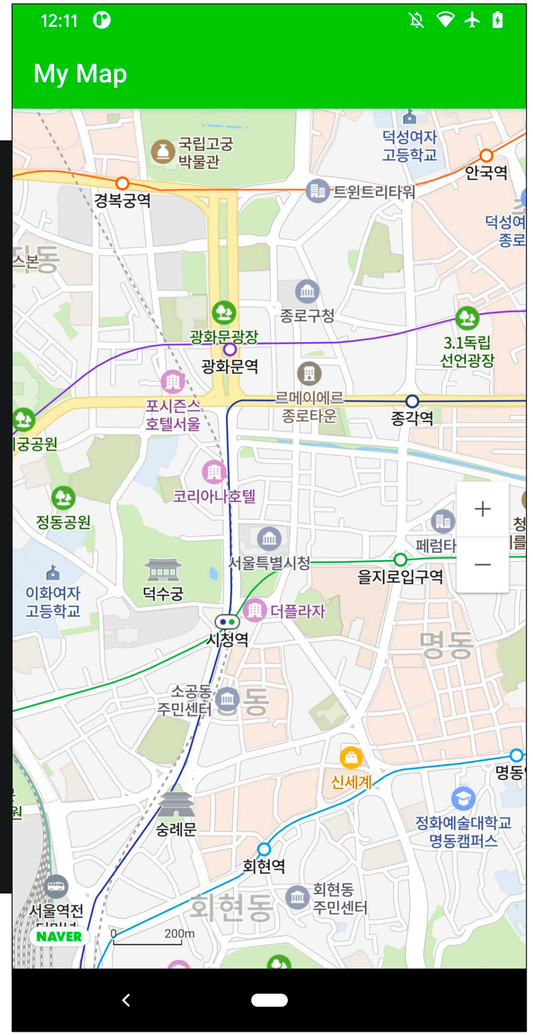
DEVIEW 2020

실행!

```xml
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <androidx.fragment.app.FragmentContainerView
        android:id="@+id/map_fragment"
        android:name="com.naver.maps.map.MapFragment"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```



3. 다양한 기능 활용

3.1 지도 옵션

DEVIEW 2020

다양한 지도 옵션을 초기에/동적으로 지정 가능

- 카메라의 위치
- 지도 유형
- 레이어
- 실내 지도
- 야간 모드
- 컨트롤 노출
- 제스처 활성화

3.1 지도 옵션

DEVIEW 2020

지도 프래그먼트를 동적으로 추가

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <androidx.fragment.app.FragmentContainerView
        android:id="@+id/map_fragment"
        android:name="com.naver.maps.map.MapFragment"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

3.1 지도 옵션

DEVIEW 2020

지도 프래그먼트를 동적으로 추가

```
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)

        setContentView(R.layout.activity_main)

        val mapFragment = supportFragmentManager.findFragmentById(R.id.map_fragment) as? MapFragment
            ?: MapFragment.newInstance().also {
                supportFragmentManager.commit {
                    replace(R.id.map_fragment, it)
                }
            }
    }
}
```

3.1 지도 옵션

DEVIEW 2020

NaverMapOptions를 이용해 초기 옵션 지정

```
val mapFragment = supportFragmentManager.findFragmentById(R.id.map_fragment) as? MapFragment
    ?: MapFragment.newInstance(
        NaverMapOptions()
            .camera(CameraPosition(LatLng(37.5116620, 127.0594274), 16.0))
            .indoorEnabled(true)
    ).also {
        supportFragmentManager.commit {
            replace(R.id.map_fragment, it)
        }
    }
```

3.1 지도 옵션

DEVIEW 2020

NaverMapFragment.getMapAsync()로 NaverMap 객체를 얻음

```
val mapFragment = supportFragmentManager.findFragmentById(R.id.map_fragment) as? MapFragment
    ?: MapFragment.newInstance(
        NaverMapOptions()
            .camera(CameraPosition(LatLng(37.5116620, 127.0594274), 16.0))
            .indoorEnabled(true)
    ).also {
        supportFragmentManager.commit {
            replace(R.id.map_fragment, it)
        }
    }

mapFragment.getMapAsync { naverMap: NaverMap ->
    // TODO
}
```

3.1 지도 옵션

DEVIEW 2020

NaverMap의 각종 메서드를 호출해 동적으로 옵션 변경

```
mapFragment.getMapAsync { naverMap ->
    fab.setOnClickListener {
        naverMap.mapType = if (naverMap.mapType == NaverMap.MapType.Basic) {
            NaverMap.MapType.Hybrid
        } else {
            NaverMap.MapType.Basic
        }
    }
}
```

3.2 이벤트

DEVIEW 2020

사용자의 터치에 반응하는 다양한 UI 이벤트 제공

- 클릭
- 롱 클릭
- 더블 탭
- 투 핑거 탭

3.2 이벤트

DEVIEW 2020

지도상의 다양한 객체가 이벤트를 받아 처리할 수 있음

- 지도 자체
- 심벌
- 오버레이

3.2 이벤트

DEVIEW 2020

지도 클릭 이벤트

```
naverMap.setOnMapClickListener { point, coord ->
    Toast.makeText(
        this,
        "지도 클릭\n${coord.latitude}\n${coord.longitude}",
        Toast.LENGTH_SHORT
    ).show()
}

naverMap.setOnMapLongClickListener { point, coord ->
    Toast.makeText(
        this,
        "지도 롱 클릭\n${coord.latitude}\n${coord.longitude}",
        Toast.LENGTH_SHORT
    ).show()
}
```

3.2 이벤트

DEVIEW 2020

심벌 클릭 이벤트

```
naverMap.setOnSymbolClickListener { symbol ->
    Toast.makeText(
        this,
        "${symbol.caption}\n${symbol.position.latitude}\n${symbol.position.longitude}",
        Toast.LENGTH_SHORT
    ).show()
    true
}
```

3.3 마커

DEVIEW 2020

- 지도에서 가장 널리 사용되는 요소
- 지도상의 한 지점을 나타내기 위한 오버레이
- 특정 좌표에 아이콘과 캡션을 표시
- 다양한 겹침 처리 옵션 제공
- 클릭 이벤트를 받아 이벤트를 소비하거나 지도로 전파할 수 있음

3.3 마커

지도 롱 클릭 또는 심벌 클릭시 마커 추가

```
fun showMarker(coord: LatLng, caption: String = "${coord.latitude}\n${coord.longitude}") {
    Marker().apply {
        position = coord
        captionText = caption
        map = naverMap
    }
}

naverMap.setOnMapLongClickListener { point, coord ->
    showMarker(coord)
}

naverMap.setOnSymbolClickListener { symbol ->
    showMarker(symbol.position, symbol.caption)
    true
}
```

3.3 마커

DEVIEW 2020

하나의 마커를 재사용

```
val marker = Marker().apply {
    icon = MarkerIcons.BLUE
    isHideCollidedSymbols = true
    setOnClickListener {
        map = null
        true
    }
}

fun showMarker(coord: LatLng, caption: String = "${coord.latitude}\n${coord.longitude}") {
    marker.apply {
        position = coord
        captionText = caption
        map = naverMap
    }
}
```

3.4 위치

- 지도를 사용하는 앱은 사용자의 위치 또한 사용하는 경우가 많음
- 사용자의 위치를 지도에 나타내고 해당 위치로 이동 등
- 이런 기능을 손쉽게 구현할 수 있도록 빌트인으로 제공

3.4 위치

DEVIEW 2020

AndroidManifest.xml에 위치 권한 선언

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.example.mymap">

    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />

    <application>

        <!-- ... -->

    </application>

</manifest>
```

3.4 위치

build.gradle에 play-services-location 의존성 추가

```
dependencies {
    implementation 'com.naver.maps:map-sdk:3.10.0'
    implementation 'com.google.android.gms:play-services-location:17.1.0'
}
```

3.4 위치

FusedLocationSource 및 권한 처리 코드 추가

```
class MainActivity : AppCompatActivity() {
    private val fusedLocationSource = FusedLocationSource(this, 100)

    override fun onRequestPermissionsResult(
        requestCode: Int,
        permissions: Array<out String>,
        grantResults: IntArray
    ) {
        if (fusedLocationSource.onRequestPermissionsResult(requestCode, permissions, grantResults)) {
            return
        }
        super.onRequestPermissionsResult(requestCode, permissions, grantResults)
    }

    // ...
}
```

3.4 위치

DEVIEW 2020

위치 버튼 추가 및 지도에 FusedLocationSource 지정

```
val mapFragment = supportFragmentManager.findFragmentById(R.id.map_fragment) as? MapFragment
    ?: MapFragment.newInstance(
        NaverMapOptions()
            .locationButtonEnabled(true)
            .indoorEnabled(true)
    ).also {
        supportFragmentManager.commit {
            replace(R.id.map_fragment, it)
        }
    }

mapFragment.getMapAsync { naverMap: NaverMap ->
    naverMap.locationSource = fusedLocationSource
    // ...
}
```

3.4 위치

나침반(지자기 센서) 기능 사용

```
naverMap.locationSource = fusedLocationSource

naverMap.addOnOptionChangeListener {
    val mode = naverMap.locationTrackingMode
    val compassEnabled = mode == LocationTrackingMode.Follow || mode == LocationTrackingMode.Face
    fusedLocationSource.isCompassEnabled = compassEnabled
}
```

3.5 카메라

- 지도는 화면 건너편의 큰 지도를 카메라로 바라보는 방식으로 표현됨
- 카메라를 움직이면 화면에 보이는 지도가 변경됨
- 이동, 확대 및 축소, 기울임, 회전 등 다양한 움직임 가능
- 목표 지점, 애니메이션, 콜백 등 다양한 옵션을 지정할 수 있음

3.5 카메라

DEVIEW 2020

마커 클릭시 마커의 위치로 카메라 이동

```
val marker = Marker().apply {
    icon = MarkerIcons.BLUE
    isHideCollidedSymbols = true
    setOnClickListener {
        val cameraUpdate = CameraUpdate.scrollTo(position).animate(CameraAnimation.Easing)
        naverMap.moveCamera(cameraUpdate)
        true
    }
}

naverMap.setOnMapClickListener { _, _ ->
    marker.map = null
}
```

3.5 카메라

DEVIEW 2020

피봇 및 콜백 사용

```
setOnClickListener {
    val cameraUpdate = CameraUpdate
        .scrollTo(position)
        .animate(CameraAnimation.Easing)
        .pivot(PointF(0.5f, 0.8f))
        .finishCallback {
            Toast.makeText(context, "카메라 이동 완료", Toast.LENGTH_SHORT).show()
        }
    naverMap.moveCamera(cameraUpdate)
    true
}
```

3.5 카메라

DEVIEW 2020

현 위치와 마커가 모두 보이는 범위로 카메라 이동

```
fab.setOnClickListener {
    if (naverMap.locationOverlay.isVisible && marker.isAdded) {
        val bounds = LatLngBounds.from(
            naverMap.locationOverlay.position,
            marker.position
        )

        naverMap.moveCamera(
            CameraUpdate
                .fitBounds(bounds, resources.getDimensionPixelSize(R.dimen.map_padding))
                .animate(CameraAnimation.Fly, 1000)
        )
    }
}
```

3.6 다양한 오버레이

- 정보 창, 셰이프, 이미지, 경로선, 화살표 등 다양한 오버레이 제공
- 모든 오버레이는 클릭 이벤트를 받을 수 있음
- 각 오버레이 간의 겹침 우선순위도 지정 가능

3.6 다양한 오버레이

현 위치와 마커를 잇는 폴리라인 추가

```
val polyline = PolylineOverlay().apply {
    width = resources.getDimensionPixelSize(R.dimen.line_width)
    color = ResourcesCompat.getColor(resources, R.color.line_color, theme)
}
fun showOverlays(coord: LatLng, caption: String = "${coord.latitude}\n${coord.longitude}") {
    // ...
    if (naverMap.locationOverlay.isVisible) {
        polyline.apply {
            coords = listOf(naverMap.locationOverlay.position, coord)
            map = naverMap
        }
    }
}
naverMap.setOnMapClickListener { _, _ ->
    marker.map = null
    polyline.map = null
}
```

3.6 다양한 오버레이

DEVIEW 2020

현 위치와 마커간의 직선거리를 정보 창으로 표시

```
val infoWindow = InfoWindow().apply {
    adapter = object : InfoWindow.DefaultTextAdapter(context) {
        override fun getText(infoWindow: InfoWindow): CharSequence = tag as? String ?: ""
    }
}

naverMap.setOnMapClickListener { _, _ ->
    marker.map = null
    polyline.map = null
    infoWindow.close()
}
```

3.6 다양한 오버레이

DEVIEW 2020

현 위치와 마커간의 직선거리를 정보 창으로 표시

```
fun showOverlays(coord: LatLng, caption: String = "${coord.latitude}\n${coord.longitude}") {
    // ...
    if (naverMap.locationOverlay.isVisible) {
        // ...
        infoWindow.apply {
            tag = "${naverMap.locationOverlay.position.distanceTo(marker.position).roundToInt()}m"
            invalidate()
            open(marker)
        }
    }
}
```

3.6 다양한 오버레이

폴리라인 대신 경로선 사용

```
val path = PathOverlay().apply {
    width = resources.getDimensionPixelSize(R.dimen.line_width)
    color = ResourcesCompat.getColor(resources, R.color.line_color, theme)
    outlineColor = ResourcesCompat.getColor(resources, R.color.outline_color, theme)
    patternImage = OverlayImage.fromResource(R.drawable.path_pattern)
}

fun showOverlays(coord: LatLng, caption: String = "${coord.latitude}\n${coord.longitude}") {
    // ...
    if (naverMap.locationOverlay.isVisible) {
        // ...
        path.apply {
            coords = listOf(naverMap.locationOverlay.position, coord)
            map = naverMap
        }
    }
}
```

4. 네이버 지도 API 연동

4.1 Maps API와 연동

지도 SDK 외에도 여러 지도 관련 API를 오픈 API로 제공하고 있음

- Geocoding(주소 → 좌표)
- Reverse Geocoding(좌표 → 주소)
- Directions(자동차 길찾기)

4.1 Maps API와 연동

DEVIEW 2020

Reverse Geocoding

- https://apidocs.ncloud.com/ko/ai-naver/maps_reverse_geocoding/
- 좌표에 해당하는 주소 반환
- 월 300만 호출 무료

4.1 Maps API와 연동

DEVIEW 2020

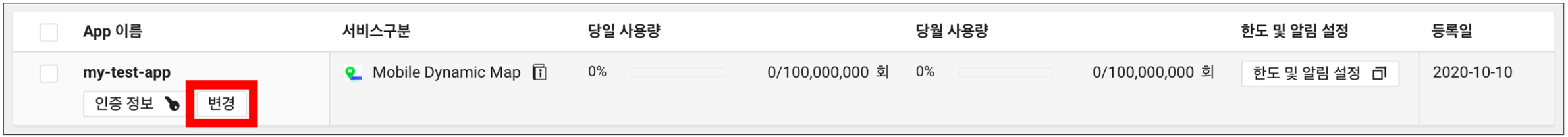
Directions 5

- https://apidocs.ncloud.com/ko/ai-naver/maps_directions/
- 자동차 길찾기 결과 반환
- 월 6만 / 일 6천 호출 무료
- 경유지 5개 지원, 15개를 지원하는 Directions 15 별도 존재

4.1 Maps API와 연동

DEVIEW 2020

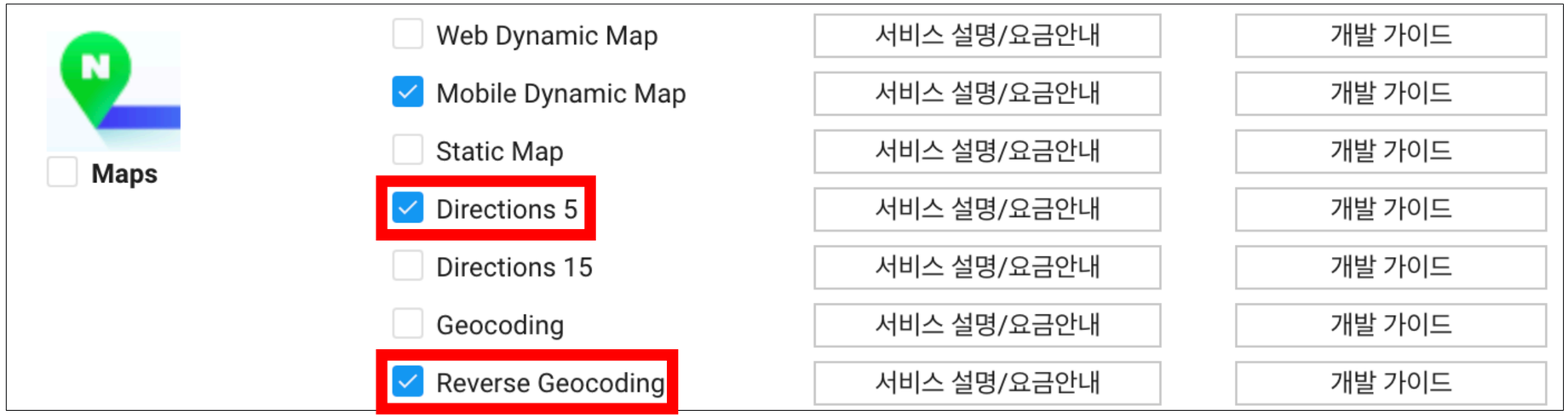
Recently Viewed → AI·NAVER API → 변경 선택



4.1 Maps API와 연동

DEVIEW 2020

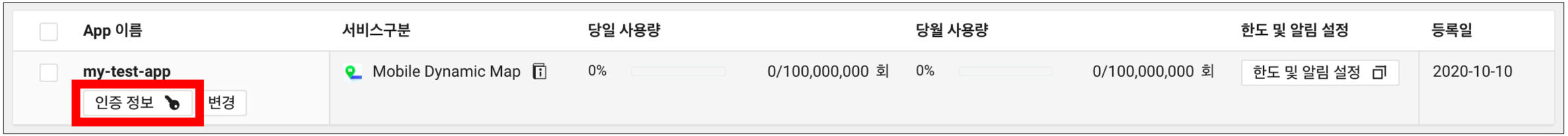
Directions 5 및 Reverse Geocoding 체크



4.1 Maps API와 연동

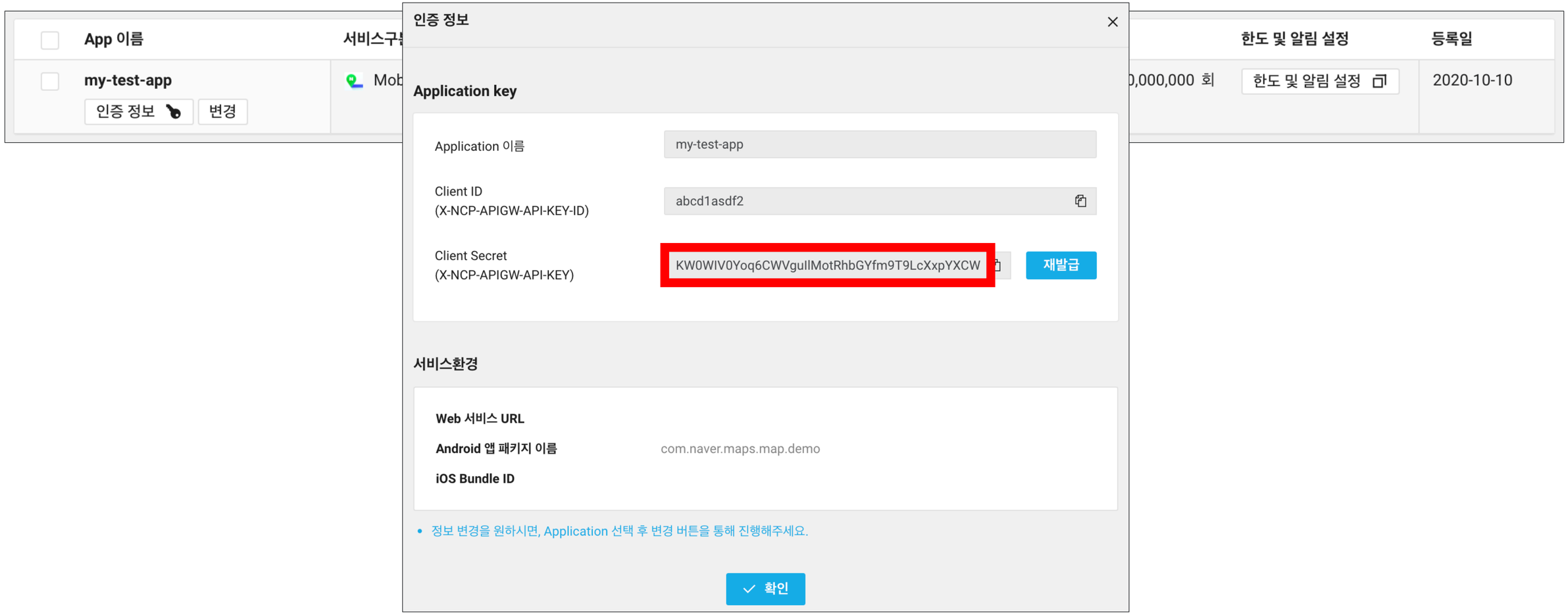
DEVIEW 2020

클라이언트 시크릿 확인



4.1 Maps API와 연동

클라이언트 시크릿 확인



4.1 Maps API와 연동

뼈대 프로젝트에서 기본적인 구현이 되어 있는 유틸리티 클래스 제공

```
class App : Application() {
    override fun onCreate() {
        super.onCreate()

        MapsApi.init(
            getString(R.string.naver_map_sdk_client_id),
            getString(R.string.naver_map_sdk_client_secret)
        )

        // ...
    }
}
```

4.1 Maps API와 연동

client_secret.xml에 발급받은 클라이언트 시크릿 입력

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="naver_map_sdk_client_secret" translatable="false">YOUR_CLIENT_SECRET_HERE</string>
</resources>
```

4.1 Maps API와 연동

DEVIEW 2020

사용 라이브러리 / 환경

- Retrofit 2
- Moshi(codegen)
- Kotlin Coroutines

4.1 Maps API와 연동

build.gradle에 사용 라이브러리 의존성 / 환경 추가

```
apply plugin: 'kotlin-kapt'

// ...

dependencies {
    implementation 'org.jetbrains.kotlinx:kotlinx-coroutines-android:1.3.9'

    implementation 'com.squareup.moshi:moshi:1.9.2'
    kapt 'com.squareup.moshi:moshi-kotlin-codegen:1.9.2'

    implementation 'com.squareup.retrofit2:converter-moshi:2.9.0'

    // ...
}
```

4.1 Maps API와 연동

DEVIEW 2020

Reverse Geocoding API 사용

```
interface ReverseGeocodingService {
    @GET("map-reversegeocode/v2/gc?output=json")
    suspend fun gc(
        @Header("X-NCP-APIGW-API-KEY-ID") clientId: String,
        @Header("X-NCP-APIGW-API-KEY") clientSecret: String,
        @Query("coords") coord: String, // "$longitude,$latitude" 포맷
    ): ReverseGeocodingResponse
}
```

4.1 Maps API와 연동

Reverse Geocoding API 사용

```
class Area(val name: String? = null)

class Region(
    val area0: Area? = null, val area1: Area? = null, val area2: Area? = null,
    val area3: Area? = null, val area4: Area? = null,
) {
    override fun toString() = listOfNotNull(area1, area2, area3, area4)
        .mapNotNull { it.name }
        .filter { it.isNotEmpty() }
        .joinToString(" ")
}

class Geocode(val region: Region? = null)

class ReverseGeocodingResponse(val results: List<Geocode> = emptyList()) {
    override fun toString() = results.firstOrNull()?.region?.toString() ?: "No Address"
}
```

4.1 Maps API와 연동

DEVIEW 2020

Retrofit 선언

```
val retrofit = Retrofit.Builder()
    .baseUrl("https://naveropenapi.apigw.ntruss.com")
    .addConverterFactory(MoshiConverterFactory.create())
    .build()

val reverseGeocodingService = retrofit.create<ReverseGeocodingService>()
```

4.1 Maps API와 연동

DEVIEW 2020

지도 롱 클릭 시 마커 캡션에 좌표 대신 주소 표시

```
fun showOverlays(coord: LatLng, caption: String? = null) {
    marker.apply {
        if (caption == null) {
            captionText = ""
            lifecycleScope.launch {
                captionText = MapsApi.reverseGeocode(coord).toString()
            }
        } else {
            captionText = caption
        }
        // ...
    }
    // ...
}
```

4.1 Maps API와 연동

DEVIEW 2020

Directions API 사용

```
interface DirectionsService {
    @GET("map-direction/v1/driving")
    suspend fun get(
        @Header("X-NCP-APIGW-API-KEY-ID") clientId: String,
        @Header("X-NCP-APIGW-API-KEY") clientSecret: String,
        @Query("start") start: String, // "$longitude,$latitude" 포맷
        @Query("goal") goal: String, // "$longitude,$latitude" 포맷
    ): DirectionsResponse
}
```

4.1 Maps API와 연동

DEVIEW 2020

Directions API 사용

```
class DirectionsResponse(val route: Map<String, List<Route>>) {
    val firstRoute
        get() = route.asIterable().firstOrNull()?.value?.firstOrNull()
}

class Route(val summary: Summary, val path: List<List<Double>>) {
    val coords
        get() = path.map { LatLng(it[1], it[0]) }
}

class Summary(val distance: Int, val duration: Long)
```

4.1 Maps API와 연동

DEVIEW 2020

현 위치와 마커 간 자동차 경로, 거리 및 소요시간 표시

```
fun showOverlays(coord: LatLng, caption: String? = null) {
    // ...
    path.map = null
    infoWindow.close()

    if (naverMap.locationOverlay.isVisible) {
        lifecycleScope.launch {
            MapsApi.directions(naverMap.locationOverlay.position, coord).firstRoute?.let {
                path.coords = it.coords
                path.map = naverMap
                infoWindow.tag = it.summary
                infoWindow.invalidate()
                infoWindow.open(marker)
            }
        }
    }
}
```

4.1 Maps API와 연동

DEVIEW 2020

현 위치와 마커 간 자동차 경로, 거리 및 소요시간 표시

```
val infoWindow = InfoWindow().apply {
    adapter = object : InfoWindow.DefaultTextAdapter(context) {
        override fun getText(infoWindow: InfoWindow): CharSequence {
            val summary = infoWindow.tag as? Summary ?: return ""
            return "${summary.distance}m / ${summary.duration / 1000 / 60}분"
        }
    }
}
```

4.2 네이버 지도 앱과 연동

DEVIEW 2020

인텐트를 이용해 네이버 지도 앱으로 랜딩할 수 있음

- `nmap://actionPath?parameter=value&appname={YOUR_APP_NAME}` 구조의 URI
- https://docs.ncloud.com/ko/naveropenapi_v3/maps/url-scheme/url-scheme.html

4.2 네이버 지도 앱과 연동

타겟 API가 30 이상이라면 AndroidManifest.xml에 <queries> 추가

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.example.mymap">

    <queries>
        <package android:name="com.nhn.android.nmap" />
    </queries>

    <!-- ... -->

</manifest>
```

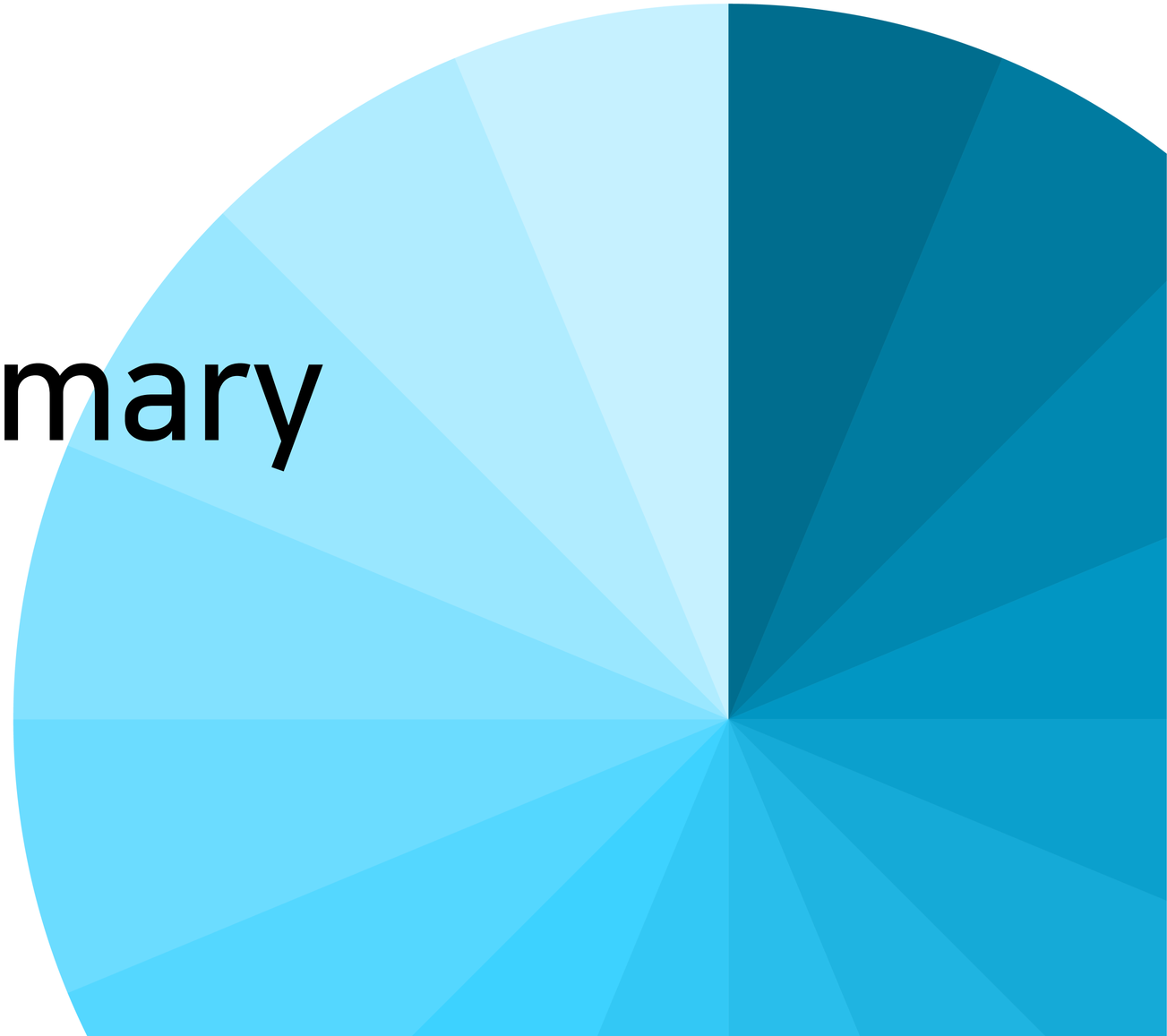
4.2 네이버 지도 앱과 연동

버튼 클릭시 네이버 지도 앱 내비게이션으로 랜딩

```
fab.setOnClickListener {
    if (naverMap.locationOverlay.isVisible && marker.isAdded) {
        val start = naverMap.locationOverlay.position
        val goal = marker.position
        val url = "nmap://navigation?appname=${BuildConfig.APPLICATION_ID}" +
                "&slat=${start.latitude}&slng=${start.longitude}&dlat=${goal.latitude}&dlng=${goal.longitude}"
        val intent = Intent(Intent.ACTION_VIEW, url.toUri()).addCategory(Intent.CATEGORY_BROWSABLE)

        context.startActivity(
            if (packageManager.queryIntentActivities(intent, PackageManager.MATCH_DEFAULT_ONLY).isEmpty()) {
                Intent(Intent.ACTION_VIEW, "market://details?id=com.nhn.android.nmap".toUri())
            } else {
                intent
            }
        )
    }
}
```

Summary



Summary

DEVIEW 2020

- 네이버 지도 SDK를 이용하면 앱에 지도를 쉽게 넣을 수 있음
- 사용량에 제한이 없으므로 부담없이 사용 가능
- 다양하고 강력한 기능을 제공하므로 복잡한 지도 앱도 만들 수 있음
- SDK 이외에도 서버 API로 제공되는 오픈 API 활용 가능

Summary

DEVIEW 2020

- https://navermaps.github.io/android-map-sdk
- https://github.com/navermaps/android-map-sdk
- https://www.ncloud.com/product/applicationService/maps

Q & A

Thank You